Churn Prediction Project Documentation

This documentation provides a comprehensive overview of the steps undertaken in building a churn prediction model using machine learning techniques. The goal of the project is to predict customer churn for a telecommunications company and create a user-friendly web application for predictions. The document outlines the entire workflow, from data preprocessing to model selection and deployment.

Step 1: Data Exploration
- Imported necessary libraries: `pandas`, `numpy`, `matplotlib`, `seaborn`.
- Loaded the customer churn dataset from an Excel file using `pd.read_excel()`.
- Inspected the first few rows of the dataset using `data.head()` to understand the structure of the data.
- Checked for duplicated records using `data.duplicated()` and removed them using `data.drop_duplicates()`.
- Handled missing values by dropping rows with missing values using `data.dropna(inplace=True)`.
- Visualized the data distribution and relationships between features using `sns.pairplot(data, hue='Churn')`.

Step 2: Data Preprocessing
- Dropped unnecessary columns (`CustomerID`, `Name`) from the dataset using `data.drop()`.
- Encoded categorical variables (`Gender`, `Location`) using `LabelEncoder` to convert them into numerical values.
- Computed the correlation matrix of numerical columns and visualized it using a heatmap to identify relationships between features.
- Detected outliers using the z-score method and stored the results in the `outliers` variable.
- Created age groups based on the `Age` feature and subscription duration groups based on the `Subscription_Length_Months` feature using the `pd.cut()` function.

Step 3: Feature Engineering
- Engineered a new feature `Total_Money_Spent` by combining the `Monthly_Bill` and `Total_Usage_GB` features.
- Created age groups based on the `Age` feature using predefined bins and labels.
- Created subscription duration groups based on the `Subscription_Length_Months` feature using predefined bins and labels.
- Dropped the original `Age` and `Subscription_Length_Months` features after engineering new features.

Step 4: Data Splitting and Model Training
- Split the preprocessed data into features (X) and the target variable (y) using `X = data.drop('Churn', axis=1)` and `y = data['Churn']`.
- Split the data into training and testing sets using `train_test_split()` from `sklearn.model_selection`.
- Used `StandardScaler()` to standardize numerical features for better model performance.
- Applied Label Encoding to categorical columns and standardized features using `StandardScaler` to prepare them for model training.

Step 5: Model Training and Evaluation
- Trained multiple individual machine learning models:
  - `Logistic Regression`
  - `Random Forest`
  - `XGBoost`
  - `SVM`
  - `KNN`
- Created an ensemble model using a `VotingClassifier` that combines the above models.
- Trained and evaluated each model using metrics like accuracy, precision, recall, and F1-score.
- Generated a results table with model names and corresponding evaluation metrics.

Step 6: Best Model Selection
- Identified the `XGBoost` model as the best-performing model due to consistently superior performance.
- Selected `XGBoost` as the final model for customer churn prediction.

Step 7: Model Hyperparameter Tuning
- Conducted a hyperparameter search using `GridSearchCV` to find optimal parameters for the `XGBoost` model.
- Defined a parameter grid with various hyperparameter values to test.
- Performed a grid search using `GridSearchCV` on the training data and evaluated the best parameters and score.

Step 8: Cross-Validation
- Used cross-validation to assess the generalization performance of the `XGBoost` model.
- Calculated cross-validation scores for accuracy using `cross_val_score()` from `sklearn.model_selection`.

Step 9: Threshold Analysis
- Analyzed model performance at different probability thresholds using the `confusion_matrix` and `f1_score`.
- Calculated sensitivity, specificity, accuracy, and F1-score for each threshold.
- Plotted sensitivity, specificity, accuracy, and F1-score against probability thresholds using `matplotlib`.

Step 10: Web Application Deployment
- Developed a web application using the `Flask` framework to interact with the trained model.
- Loaded the trained `XGBoost` model and `StandardScaler` for feature scaling from pickled files.
- Created HTML templates for the user interface using the Jinja2 template engine.
- Defined routes for the home page and prediction using `@app.route()` decorators.
- Extracted user inputs, standardized them, and made predictions using the model.
- Displayed the prediction result on the web page.
- Provided instructions on how to use the web application for churn predictions.

Conclusion:
- Successfully built a churn prediction model using machine learning techniques.
- Implemented a user-friendly web application for easy customer churn predictions.
- Demonstrated the importance of data preprocessing, feature engineering, model selection, and evaluation.
- Highlighted the potential benefits of churn prediction for businesses in customer retention efforts.

This detailed documentation captures every step of the churn prediction project, from data preprocessing to model training, evaluation, deployment, and conclusion. It showcases the entire workflow, key decisions made, and the rationale behind each choice.

extra- inorder to use the deployment of the project --

nstall Required Libraries:

In your terminal or command prompt, navigate to project directory and install the necessary libraries using the following command:

pip install flask xgboost scikit-learn
flask: Install Flask, the web framework that will help us build the application.
xgboost: Install the XGBoost library for loading the trained model.
scikit-learn: Install scikit-learn for preprocessing and model evaluation.

after install the Required Libraries

run -
"python deploy.py" in cmd